LAB Logbook

Lab 1

Lab Logbook Requirement:

1) Create a vector using np.arange.

Determine the number of the vector elements using the following method: Take the last two digits from your SID. It should be from 00 to 99. If this number is 10 or more, it becomes the required number of the vector elements. If it is less than 10, add 100 to your number.

For example, if your SID is 2287467, and the last two digits are 67, which is greater than 10. The required number is 67. If your SID is 2287407, and the last two digits are 07, which is less than 10. The required number is 107.

Then,

2. Change matrix a to 2-d array with 1 row. Print the array. You should have the two sets of brackets for a 2-d array with one row.
3. Save it in another array. Print the array.
4. Check the shape attribute value.
5. Add the code and result to your Lab Logbook

NOTE: DON'T FORGET TO SAVE AND BACK UP YOUR COMPLETED JUPYTER NOTEBOOK AND LAB LOGBOOK ON GITHUB OR ONEDRIVE.

In [69]:
```
SID = 2285050
```

In [72]:
```
number_of_vector_elements = 50
vector = np.arange(number_of_vector_elements)
print("Vector: ", vector)
```

```
Vector:  [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49]
```

In [73]:
```
_2d_array = vector.reshape(1, -1)
print("2D array with one row: ", _2d_array)
```

```
2D array with one row:  [[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
  24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
  48 49]]
```

In [74]:
```
another_array = _2d_array.copy()
print("Another array: ", another_array)
```

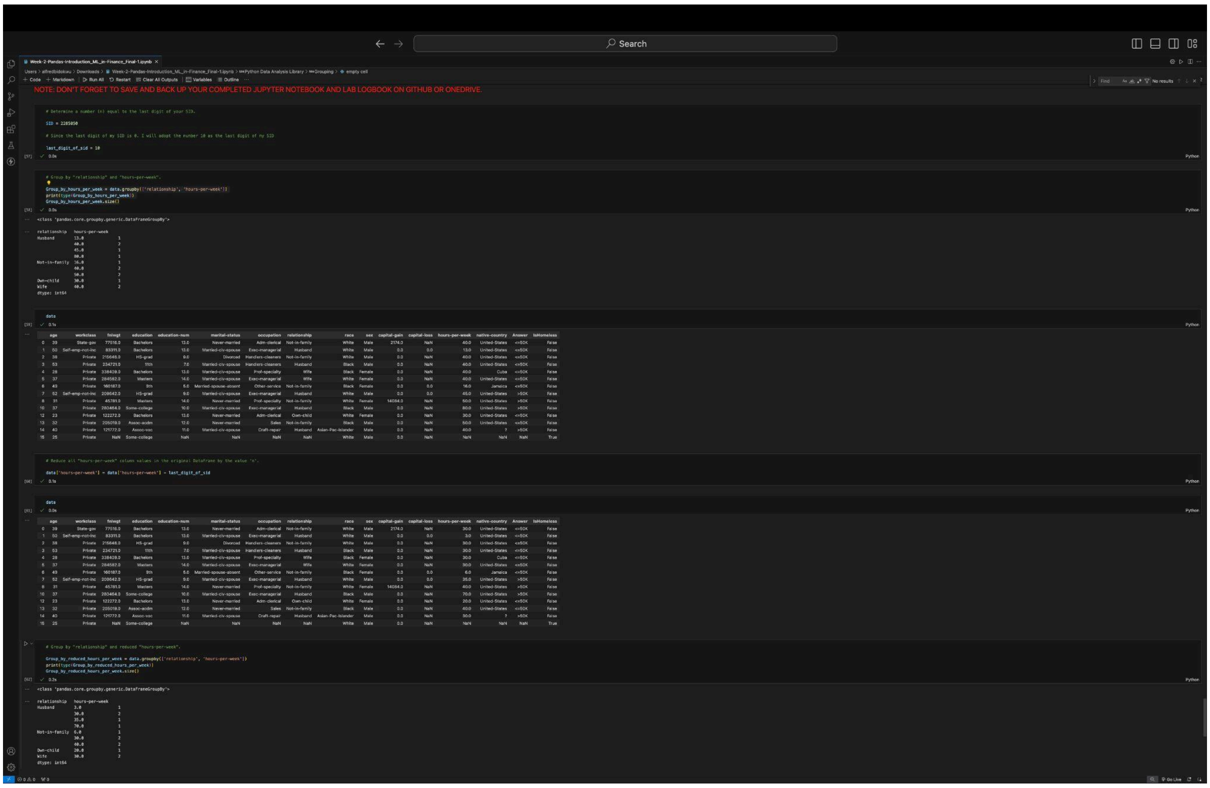
```
Another array:  [[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
  24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
  48 49]]
```

In [75]:
```
shape_value = another_array.shape
print("Shape: ", shape_value)
```

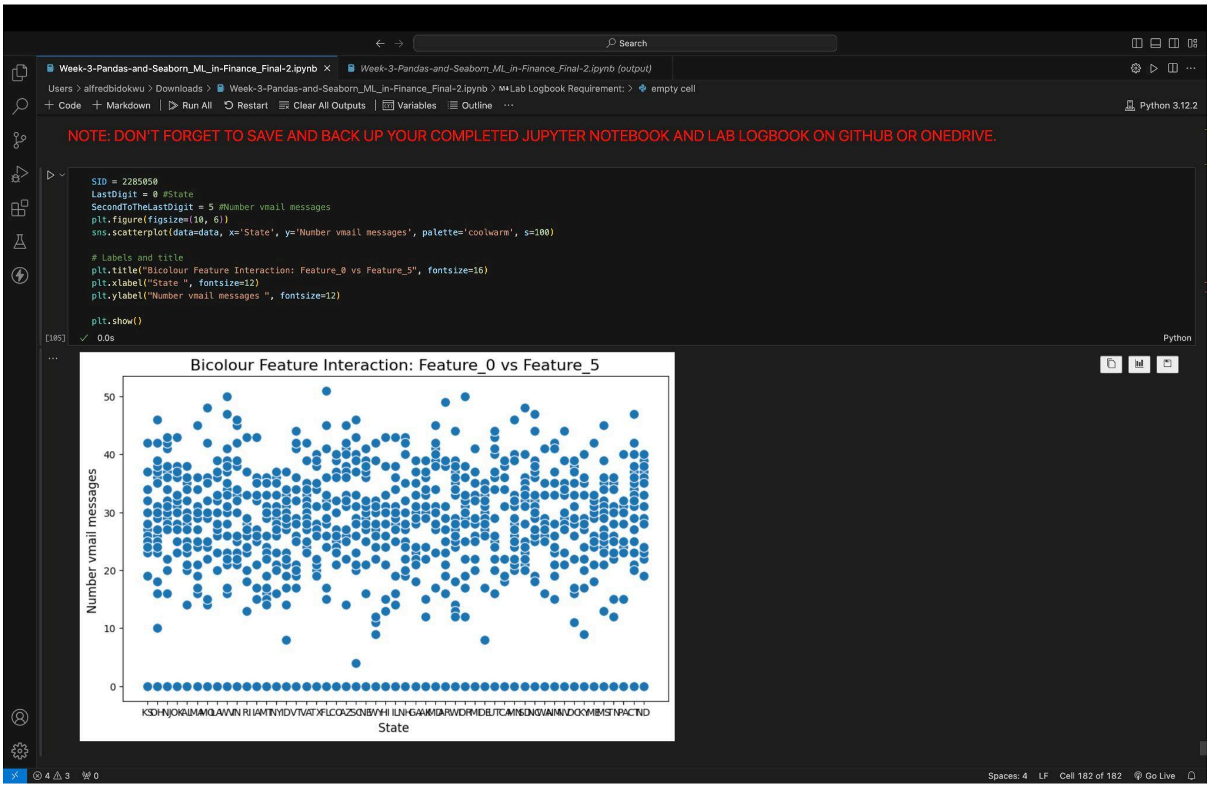
```
Shape:  (1, 50)
```

In [ ]:

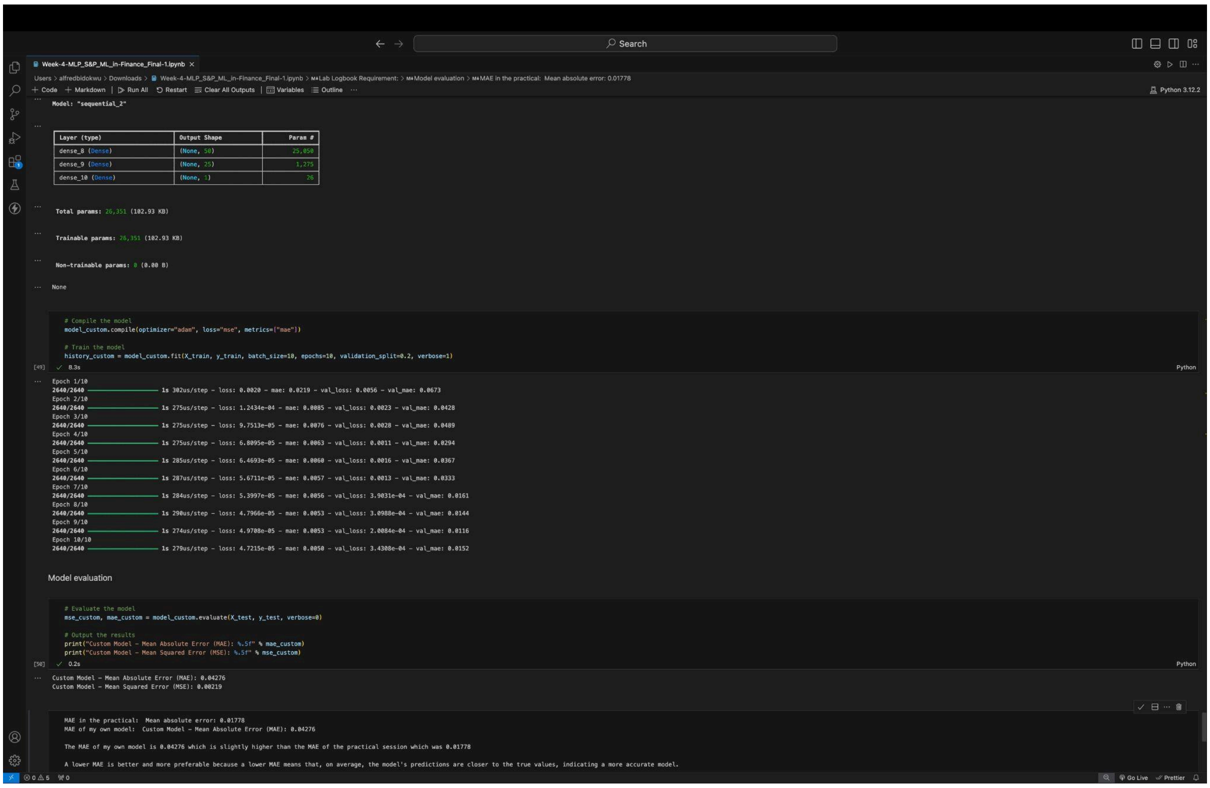
Lab 2



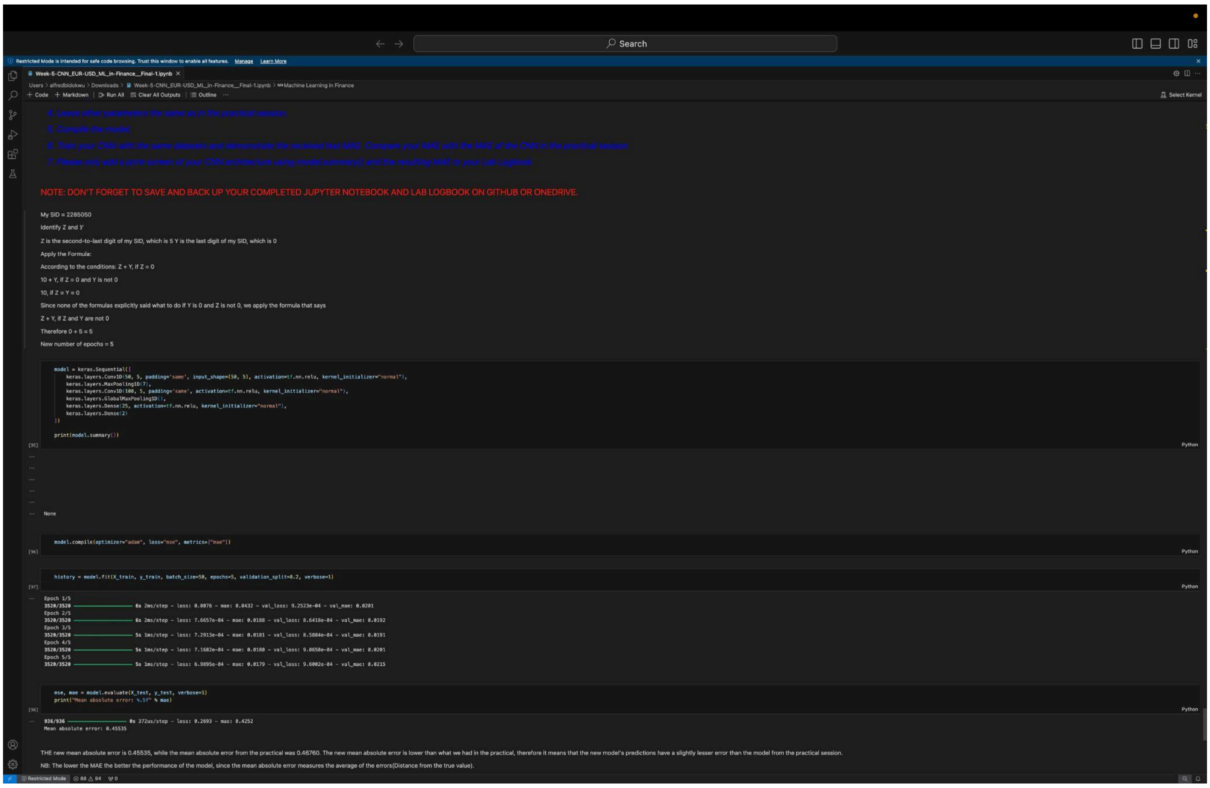
Lab 3


Week-3-Pandas-and-Seaborn_ML_in-Finance_Final-2.ipynb
NOTE: DON'T FORGET TO SAVE AND BACK UP YOUR COMPLETED JUPYTER NOTEBOOK AND LAB LOGBOOK ON GITHUB OR ONEDRIVE.
SID = 2285050
LastDigit = 0 #State
SecondToTheLastDigit = 5 #Number vmail messages
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='State', y='Number vmail messages', palette='coolwarm', s=100)
# Labels and title
plt.title("Bicolour Feature Interaction: Feature_0 vs Feature_5", fontsize=16)
plt.xlabel("State ", fontsize=12)
plt.ylabel("Number vmail messages ", fontsize=12)
plt.show()
Bicolour Feature Interaction: Feature_0 vs Feature_5
Number vmail messages
State

Lab 4


Week-4-MLP_S&P_ML_in-Finance_Final-1.ipynb
Model: "sequential_2"
Layer (type) | Output Shape | Param #
dense_8 (Dense) | (None, 50) | 25,050
dense_9 (Dense) | (None, 25) | 1,275
dense_10 (Dense) | (None, 1) | 26
Total params: 26,351 (102.93 KB)
Trainable params: 26,351 (102.93 KB)
Non-trainable params: 0 (0.00 B)
None
# Compile the model
model_custom.compile(optimizer="adam", loss="mse", metrics=["mae"])
# Train the model
history_custom = model_custom.fit(X_train, y_train, batch_size=10, epochs=10, validation_split=0.2, verbose=1)
Epoch 1/10
2640/2640 1s 302us/step - loss: 0.0020 - mae: 0.0219 - val_loss: 0.0056 - val_mae: 0.0673
Epoch 2/10
2640/2640 1s 275us/step - loss: 1.2434e-04 - mae: 0.0085 - val_loss: 0.0023 - val_mae: 0.0428
Epoch 3/10
2640/2640 1s 275us/step - loss: 9.7513e-05 - mae: 0.0076 - val_loss: 0.0028 - val_mae: 0.0489
Epoch 4/10
2640/2640 1s 275us/step - loss: 6.8095e-05 - mae: 0.0063 - val_loss: 0.0011 - val_mae: 0.0294
Epoch 5/10
2640/2640 1s 285us/step - loss: 6.4693e-05 - mae: 0.0060 - val_loss: 0.0016 - val_mae: 0.0367
Epoch 6/10
2640/2640 1s 287us/step - loss: 5.6711e-05 - mae: 0.0057 - val_loss: 0.0013 - val_mae: 0.0333
Epoch 7/10
2640/2640 1s 284us/step - loss: 5.3997e-05 - mae: 0.0056 - val_loss: 3.9031e-04 - val_mae: 0.0161
Epoch 8/10
2640/2640 1s 290us/step - loss: 4.7966e-05 - mae: 0.0053 - val_loss: 3.0988e-04 - val_mae: 0.0144
Epoch 9/10
2640/2640 1s 274us/step - loss: 4.9708e-05 - mae: 0.0053 - val_loss: 2.0084e-04 - val_mae: 0.0116
Epoch 10/10
2640/2640 1s 279us/step - loss: 4.7215e-05 - mae: 0.0050 - val_loss: 3.4308e-04 - val_mae: 0.0152
Model evaluation
Evaluate the model
mse_custom, mae_custom = model_custom.evaluate(X_test, y_test, verbose=0)
# Output the results
print("Custom Model - Mean Absolute Error (MAE): %.5f" % mae_custom)
print("Custom Model - Mean Squared Error (MSE): %.5f" % mse_custom)
Custom Model - Mean Absolute Error (MAE): 0.04276
Custom Model - Mean Squared Error (MSE): 0.00219
MAE in the practical: Mean absolute error: 0.01778
MAE of my own model: Custom Model - Mean Absolute Error (MAE): 0.04276
The MAE of my own model is 0.04276 which is slightly higher than the MAE of the practical session which was 0.01778
A lower MAE is better and more preferable because a lower MAE means that, on average, the model's predictions are closer to the true values, indicating a more accurate model.

Lab 5


Week-5-CNN_EUR-USD_ML_in-Finance__Final-1.ipynb
NOTE: DON'T FORGET TO SAVE AND BACK UP YOUR COMPLETED JUPYTER NOTEBOOK AND LAB LOGBOOK ON GITHUB OR ONEDRIVE.
My SID = 2285050
Identify Z and Y
Z is the second-to-last digit of my SID, which is 5 Y is the last digit of my SID, which is 0
Apply the Formula:
According to the conditions: Z + Y, if Z = 0
10 + Y, if Z = 0 and Y is not 0
10, if Z = Y = 0
Since none of the formulas explicitly said what to do if Y is 0 and Z is not 0, we apply the formula that says
Z + Y, if Z and Y are not 0
Therefore 0 + 5 = 5
New number of epochs = 5
model = keras.Sequential([
keras.layers.Conv1D(50, 5, padding='same', input_shape=(50, 5), activation=tf.nn.relu, kernel_initializer="normal"),
keras.layers.MaxPooling1D(7),
keras.layers.Conv1D(100, 5, padding='same', activation=tf.nn.relu, kernel_initializer="normal"),
keras.layers.GlobalMaxPooling1D(),
keras.layers.Dense(25, activation=tf.nn.relu, kernel_initializer="normal"),
keras.layers.Dense(2)
])
print(model.summary())
None
model.compile(optimizer="adam", loss="mse", metrics=["mae"])
history = model.fit(X_train, y_train, batch_size=50, epochs=5, validation_split=0.2, verbose=1)
Epoch 1/5
3520/3520 6s 2ms/step - loss: 0.0076 - mae: 0.0432 - val_loss: 9.2523e-04 - val_mae: 0.0201
Epoch 2/5
3520/3520 6s 2ms/step - loss: 7.6657e-04 - mae: 0.0188 - val_loss: 8.6418e-04 - val_mae: 0.0192
Epoch 3/5
3520/3520 5s 1ms/step - loss: 7.2913e-04 - mae: 0.0181 - val_loss: 8.5884e-04 - val_mae: 0.0191
Epoch 4/5
3520/3520 5s 1ms/step - loss: 7.1682e-04 - mae: 0.0180 - val_loss: 9.0650e-04 - val_mae: 0.0201
Epoch 5/5
3520/3520 5s 1ms/step - loss: 6.9895e-04 - mae: 0.0179 - val_loss: 9.6002e-04 - val_mae: 0.0215
mse, mae = model.evaluate(X_test, y_test, verbose=1)
print("Mean absolute error: %.5f" % mae)
936/936 0s 372us/step - loss: 0.2693 - mae: 0.4252
Mean absolute error: 0.45535
THE new mean absolute error is 0.45535, while the mean absolute error from the practical was 0.46760. The new mean absolute error is lower than what we had in the practical, therefore it means that the new model's predictions have a slightly lesser error than the model from the practical session.
NB: The lower the MAE the better the performance of the model, since the mean absolute error measures the average of the errors(Distance from the true value).

Lab 6

Lab Logbook Requirement:

1. Plot the price chart of the part of the whole dataset 'High_Bid' and 'Low_Bid' prices using iplot() library.
2. The start point should equal the 5 last digits of your SID Number.
3. The time period (in minutes) should equal the 3 last digits of your SID Number.
4. Please only add a print-screen of your code and final graph to your Lab Logbook.

NOTE: DON'T FORGET TO SAVE AND BACK UP YOUR COMPLETED JUPYTER NOTEBOOK AND LAB LOGBOOK ON GITHUB OR ONEDRIVE.

My SID is 2285050

Therefore my last 5 digits are 85050 while my last 3 digits are 050

```python
data2.iloc[85050:85100][['High_Bid', 'Low_Bid']].iplot(
                                                mode='lines+markers',
                                                xTitle='Prices', yTitle='Numbers',
                                                title='GOLD ')
```

[162] ✓ 0.0s

Lab 7

```python
print(model.summary())
```

[96] ✓ 0.0s

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 100)	47,600
dense (Dense)	(None, 2)	202

Total params: 143,408 (560.19 KB)

Trainable params: 47,802 (186.73 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 95,606 (373.46 KB)

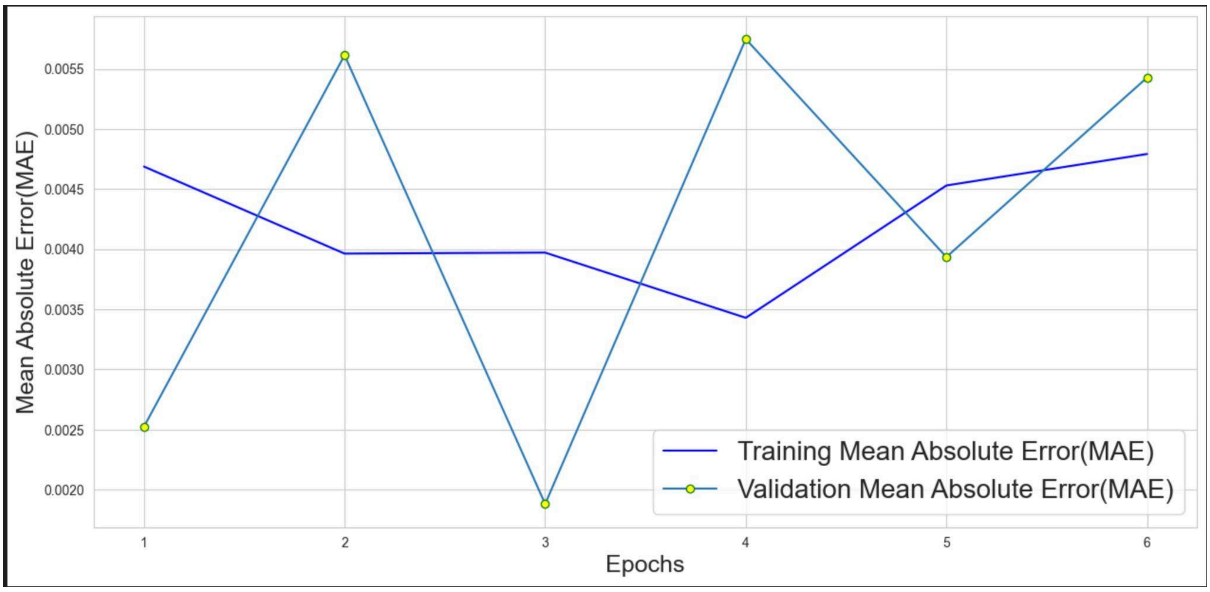
None

```
print("Mean squared error (mse): %.9f " % (scores[0]))

print("Mean absolute error (mae): %.9f " % (scores[1]))
```

[92] ✓ 0.0s

```
Mean squared error (mse): 0.000025137
Mean absolute error (mae): 0.003919592
```



Comparism between the MSE & MAE in the practical session and the MSE & MAE of the Assignment

Mean Squared Error (MSE): Practical session: 0.000052403 Assignment: 0.000025137 The MSE is significantly lower in the assignment model. MSE measures the average squared differences between predicted and actual values, which gives more weight to larger errors. The lower MSE in the assignment implies that the predictions of the assignment model are closer to the actual values, with fewer large errors compared to the practical model.

Mean Absolute Error (MAE): Practical session: 0.005791005 Assignment: 0.003919592 The MAE is also lower in the assignment model. MAE measures the average magnitude of errors in the predictions, without considering their direction. A lower MAE indicates that, on average, the assignment model's predictions deviate less from the actual values than the practical model's predictions.

Implications: Improved Model Performance: The lower MSE and MAE in the assignment model suggest that the changes in model parameters have led to improved accuracy and predictive capability. Fewer Large Errors: Since MSE is more sensitive to large errors, the lower value for the assignment model indicates that it is better at avoiding significant deviations from actual values. Refinement of Parameters: The parameter adjustments made in the assignment have likely enhanced the model's ability to capture the underlying patterns in the data more effectively.

Therefore, the assignment model is quantitatively better than the practical model, both in terms of average error magnitude (MAE) and error sensitivity (MSE).

<u>Lab 8</u>

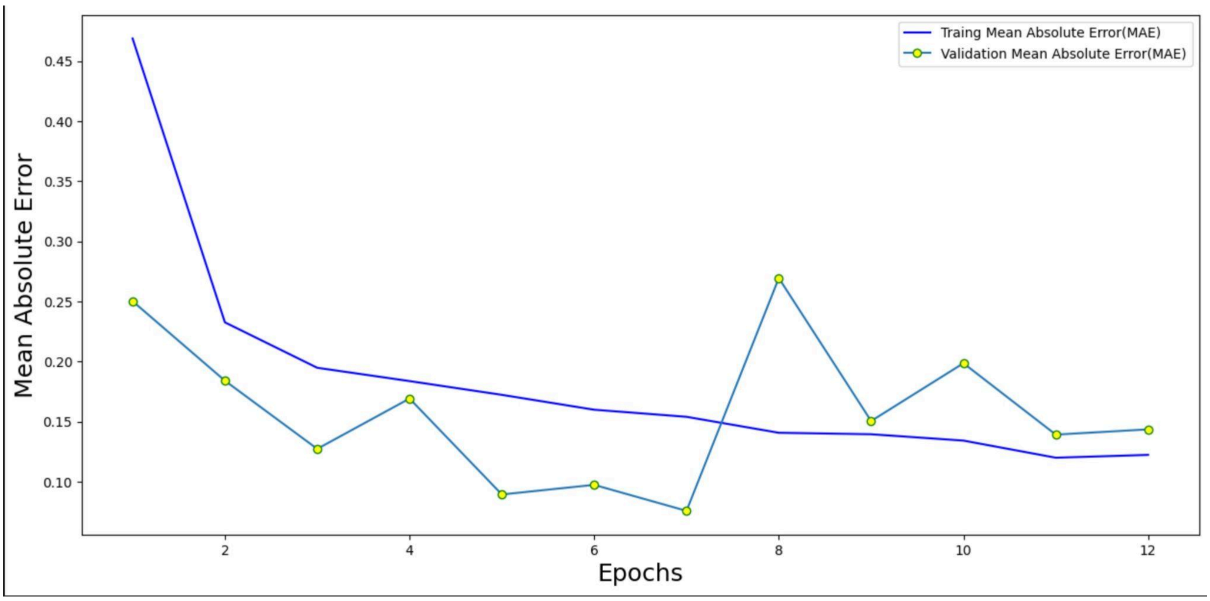
CNN METRICS:

```
print("Mean squared error (mse): %.9f " % (scores[0]))
```

```
Mean squared error (mse): 0.009019411
```

```
print("Mean absolute error (mae): %.9f " % (scores[1]))
```

```
Mean absolute error (mae): 0.073248647
```



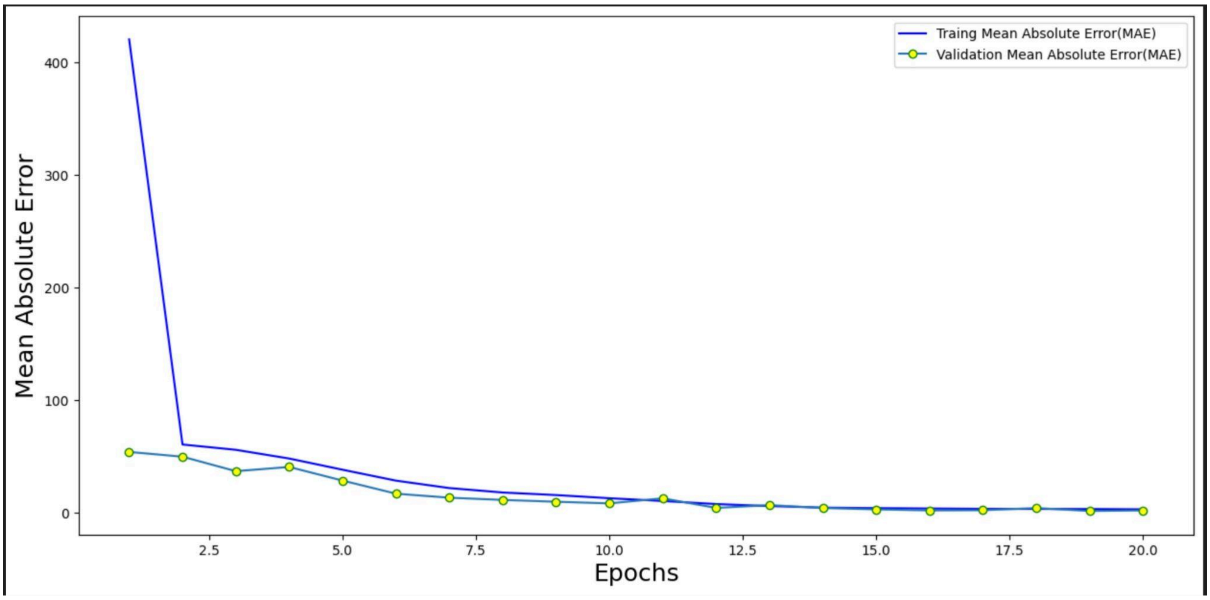
LSTM METRICS:

```
print("Mean squared error (mse): %.9f " % (scores[0]))
```

```
Mean squared error (mse): 7.449756145
```

```
print("Mean absolute error (mae): %.9f " % (scores[1]))
```

```
Mean absolute error (mae): 1.905555487
```



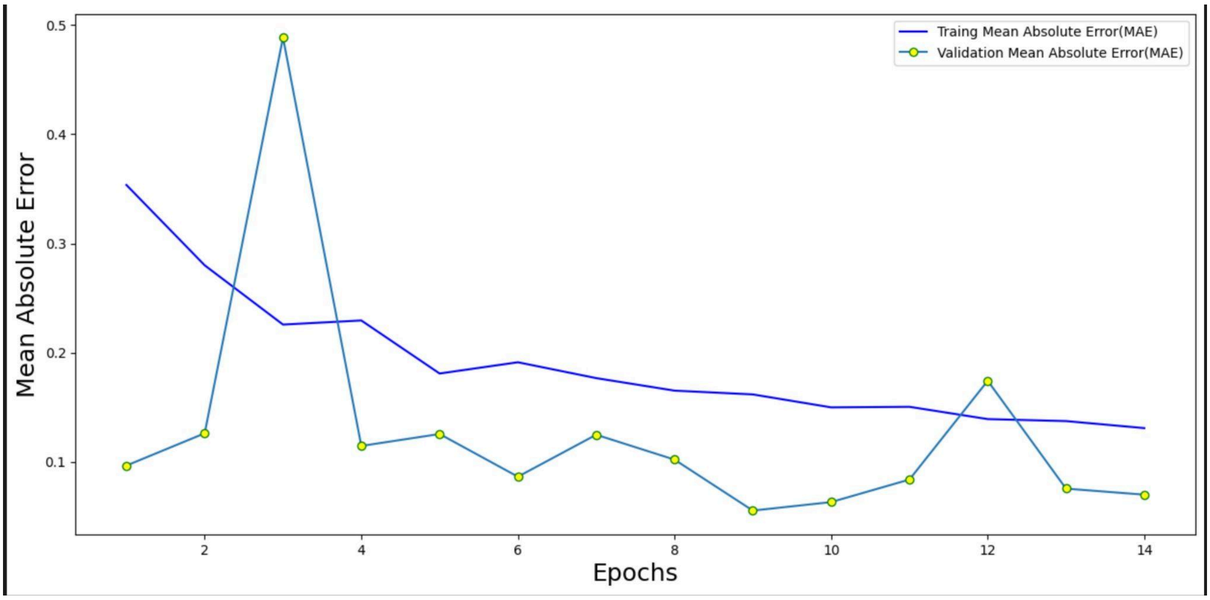
MLP METRICS:

```
print("Mean squared error (mse): %.9f " % (scores[0]))
✓ 0.0s

Mean squared error (mse): 0.005189301
```

```
print("Mean absolute error (mae): %.9f " % (scores[1]))
✓ 0.0s

Mean absolute error (mae): 0.051906992
```



Lab 9

Lab 10

Lab 11

Lab 12